2013-1378, -1414

United States Court of Appeals for the Federal Circuit

ANCORA TECHNOLOGIES, INC.,

Plaintiff- Appellant,

v.

APPLE, INC.,

Defendant-Cross Appellant.

Appeals From The United States District Court
For The Northern District Of California In
11-CV-6357, Judge Yvonne Gonzales Rogers

**APPEAL BRIEF OF PLAINTIFF-APPELLANT,
ANCORA TECHNOLOGIES, INC.**

Mark A. Cantor
John S. LeRoy
Marc Lorelli
John P. Rondini
BROOKS KUSHMAN P.C.
1000 Town Center
Twenty-Second Floor
Southfield, Michigan 48075-1238
(248) 358-4400

Attorneys for Plaintiff-Appellant

Dated: July 2, 2013

CERTIFICATE OF INTEREST

Counsel for Plaintiff-Appellant, Ancora Technologies, Inc., certifies the following:

1. The full name of every party or amicus represented by me is: _______ Ancora Technologies, Inc.

2. The name of the real party in interest (if the party named in the caption is not the real party in interest) represented by me is: _________________ Ancora Technologies, Inc.

3. All parent corporations and any publicly held companies that own 10 percent or more of the stock of the party or amicus curiae represented by me are: __ None

4. The names of all law firms and the partners or associates that appeared for the party or amicus now represented by me in the trial court or agency or are expected to appear in this court are:

Mark A. Cantor
John S. LeRoy
Marc Lorelli
John P. Rondini
BROOKS KUSHMAN P.C.
1000 Town Center
Twenty-Second Floor
Southfield, Michigan 48075

TABLE OF CONTENTS

TABLE OF AUTHORITIES

STATEMENT OF RELATED CASES

No appeal in or from the same civil action was previously before this Court or any other appellate court.

STATEMENT OF JURISDICTION

The United States District Court for the Northern District of California had jurisdiction over this suit for patent infringement under 28 U.S.C. §§ 1331 and 1338(a). This Court has exclusive jurisdiction over this appeal under 28 U.S.C. § 1295(a)(1). The district court entered final judgment of non-infringement against plaintiffs, and dismissed all other claims and counterclaims, on April 29, 2013. (A1-2.) Plaintiff Ancora Technologies, Inc. ("Ancora") filed a timely Notice of Appeal on April 30, 2013. (A2090-2091.)

STATEMENT OF THE ISSUE

Whether the district court erred when it narrowed the ordinary meaning of the term “program” to “application” program, to exclude operating system (“OS”) programs, where (i) the parties do not dispute that an operating system is a type of computer “program,” (ii) the specification uses the term “program” in its broadest sense, (iii) other non-asserted claims recite “application” program whereas the asserted claim recites only “program,” and (iv) the Examiner stated in the Notice of Allowance that the claimed “program” may be “running at the OS level.”

STATEMENT OF THE CASE

Ancora alleges that Apple's "iOS" operating system infringes U.S. Patent No. 6,411,941 (the '941 patent). Ancora sued Apple for infringement in December 2010 in the Central District of California. (A51-54.) A year later, the case was transferred to the Northern District of California. (A34.)

The parties submitted six claim terms for construction, including the term "program" recited in claim 1. (A87-96.) Claim 1 is the only asserted independent claim. (*Id.*) On December 31, 2012, the U.S. District Court for the Northern District of California issued a claim construction order construing the disputed terms. (A3-23.) The district court construed the term "program" to mean "software applications" which, in the district court's view, excluded an "operating system." (A13-18.)

"Based on the Court's finding that the term 'program' does not include the accused Apple's iOS operating system," the Court granted summary judgment, which Ancora did not oppose given the district court's claim construction order. (A2084-86.)

The district court entered final judgment on April 29, 2013. (A1-2.) Ancora appealed on April 30, 2013. (A2090-92.)

STATEMENT OF THE FACTS

A. The Invention Of The '941 Patent

The '941 patent discloses a novel approach to reducing software piracy. Inventor Miki Mullor ("Mullor") recognized a need for his invention while he was developing software for helping architects manage their drawings. (A1823.) He recognized the piracy problem he would likely face if his drawing management software was successful in the marketplace. (*Id.*)

Mullor considered prior art approaches to reducing software piracy. One approach included hardware "dongles" that physically attach to the back of a computer for licensing software running on the computer. (A1823-24.) While typically effective at reducing piracy, dongles are very cumbersome for the user, and expensive for the software developer to manufacture and ship to customers. (*Id.*) Mullor also considered software-based anti-piracy techniques, such as the use of software license keys (alpha-numeric codes) that are input at the time the software is installed. (A1824.) While much less expensive than dongles, the software-based approach is still a hassle for users because it requires users to keep track of the license keys, and it is less secure because the license keys can easily be shared together with pirated copies of the software. (*Id.*)

Facing these drawbacks, Mullor sought another approach to reducing piracy. (*Id.*) He sought an approach having the advantages of the known techniques, but not the disadvantages. (*Id.*) To achieve the security of the prior art dongles, but without the hassle of having to supply software users with new hardware for attaching to their computer, Mullor focused on the computers themselves to "figure out if there's anything in those computers that's already there that [he] could potentially reuse for a different purpose and have this effect of a, like a dongle that we had before, that hardware attachment, without having to send the hardware devices to my users." (*Id.*)

Mullor and his co-inventor Julian Valiko focused their attention on using the computer's already-existing "BIOS" (Basic Input Output System) to "create an elegant solution that has the advantages with none of the disadvantages" of the prior art approaches. (A1824-1825.) BIOS was a component of most computers, typically located on the computer's motherboard. (A4.) During examination, the Examiner defined "BIOS" as "the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices." (A10.)

The inventors recognized that the already-existing computer memory in which the BIOS was stored included extra storage space that was not being used. (A1830.) The inventors decided to use this extra space to store the software licensing information that was traditionally stored in the external dongles. (*Id.*) The claim construction order summarized the invention as follows.

> Patentees developed a third approach that had the advantages of both the hardware approach and software approach without the disadvantages of either. Patentees identified available memory space in hardware stored on the computer's motherboard, the BIOS, which they repurposed to store software licensing technology. The inventive aspect of the '941 Patent is that the writable, non-volatile memory[[1]] of the BIOS is not ordinarily considered to be a storage medium for software licensing technology. The advantage of using the BIOS for this purpose is that the level of programming expertise required to tamper with data stored in the BIOS is substantially greater than the expertise needed to tamper with data residing in volatile memory, and unsuccessful tampering comes with higher risk as it could render the computer inoperable.

(A4-5.)

The inventors spent 1997 constructing a prototype of their invention to prove their idea worked, and they filed their patent application in 1998. (A1825.)

B. Claim 1 Of The '941 Patent

Claim 1 is reproduced below in its entirety, with emphasis added to show the disputed term "program" in context.

[1] The district court construed the term "non-volatile memory" consistent with its ordinary meaning, as "memory whose data is maintained when the power is removed." (A9.) "Volatile memory," in contrast, is "memory whose data is not maintained when the power is removed." (*Id.*)

1. A method of restricting software operation within a license for use with a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area the method comprising the steps of:

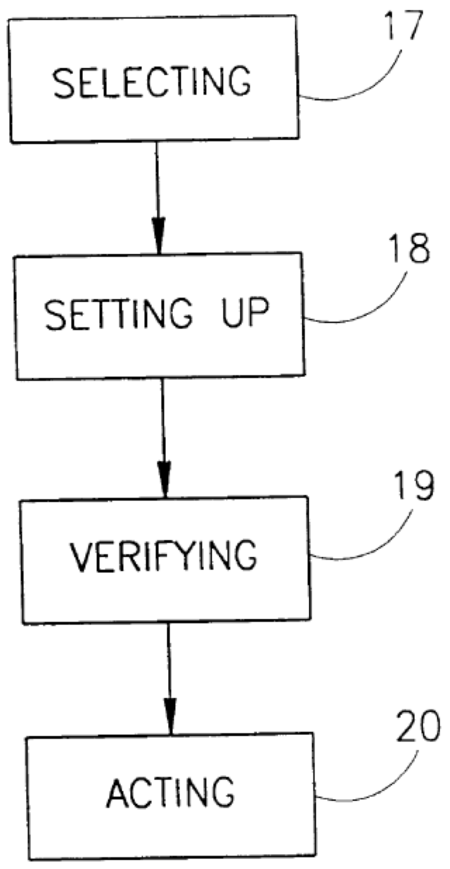
selecting a **program** residing in the volatile memory;

using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record,

verifying the **program** using at least the verification structure from the erasable non-volatile memory of the BIOS, and

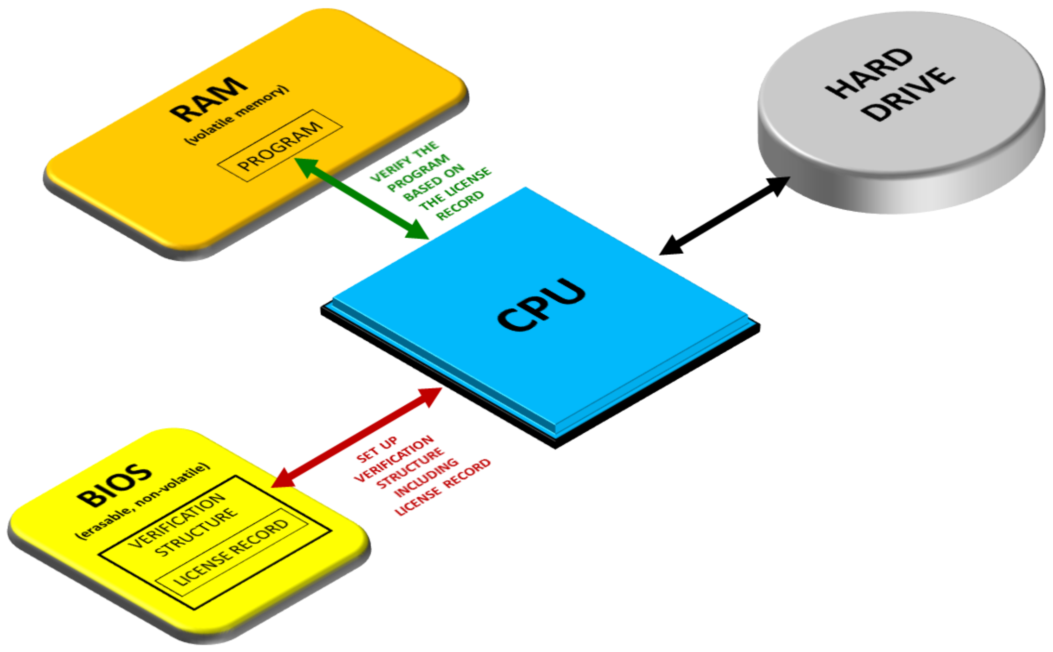
acting on the **program** according to the verification.

(A47 6:59 – A48 7:4.)



(A44, Fig. 2.)

The following graphic illustrates an example of setting up the verification structure in the BIOS, and verifying the program using the license record, as implemented in a typical computer.



In the Notice of Allowability, the Examiner summarized the Reasons for Allowance of claim 1:

> [T]he closest prior art systems, singly or collectively, do not teach licensed programs running at the OS [Operating System] level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification. Further, it is well known to those of ordinary skill of the art that a computer BIOS is not setup to manage a software license verification structure. The present invention overcomes this difficulty by using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS.

(A302, bracketed text added.)

The '941 patent issued in June 2002 (A42), and was reexamined in May 2009 (A49). The patentability of each claim was confirmed in the reexamination. (A49.)

C. The District Court's Order

As stated above, the district court construed six terms in claim 1, including the "program" term at issue on appeal. The district court construed the term "program" to mean "software applications" which, in the district court's view, excluded an "operating system." (A13-18.)

"Based on the Court's finding that the term 'program' does not include the accused Apple's iOS operating system," the Court granted summary judgment, which Ancora did not oppose given the district court's claim construction order. (A2084-86.)

SUMMARY OF THE ARGUMENT

The district court improperly narrowed the term "program" to "application," a term that the district court held excluded operating system software.

Apple does not dispute that the term "program" ordinarily encompasses an operating system. The '941 patent repeatedly uses the term "program" in the claims and the written description, but does not define the term or even mention an "operating system." The specification describes an "application" as an "example" of a "program," stating the example "is by no means binding." Confirming that the term "program" is broader than "application," the Applicant amended unasserted independent claim 18 to narrow "program" to "*application software* program." Asserted claim 1, however, was not limited to an "application" program.

The district court held that an Applicant remark in the file history gave rise to prosecution disclaimer, but that remark pertained to the software that performs the method steps – <u>not</u> the claimed "program" on which those steps operate. Contrary to the district court's decision, the Examiner stated in the Notice of Allowance that the claimed "program" may be "running at the OS level."

The intrinsic record is unanimous that the broad term "program" includes both application programs and operating system programs. The district court's contrary construction should be reversed, and the term "program" should be construed according to its ordinary meaning, namely "a set of instructions that can be executed by a computer." (A13.)

ARGUMENT

A. The District Court Erred When It Limited "Programs" to "Software Applications" And Excluded Operating System Programs

1. There Is No Dispute That An "Operating System" Is A Type Of Computer "Program"

In its *Markman* brief, Ancora provided intrinsic and extrinsic evidence establishing that those of skill in the art consider an operating system to be a type of computer "program." (A124.) The prior art Misra reference, upon which the Examiner relied in rejecting claim 1 during examination, expressly describes an "operating system" as a type of "program." (A1420, at 5:65: "programs include a server operating system.") *See, Acumed LLC v. Stryker Corp.*, 483 F.3d 800, 809 (Fed. Cir. 2007) (cited references are part of the intrinsic record). Apple's own patents confirm that an "operating system" is a "program." (A1439, U.S. Pat. No. 6,178,464 at 3:34-35: "An operating system 180 is a program that controls processing by CPU 110.").

Apple's *Markman* response brief did not dispute that those of skill in the art consider an operating system as a type of "program." Apple instead contended that fact was "irrelevant." (A1474, n. 11.)

The district court also understood the breadth of the term "program" to those of skill in the art, finding that "to a computer programmer, a program is a 'set of instructions.'' (A14.)

2. The '941 Specification Does Not Limit "Program" To "Application" Software Nor Exclude Operating Systems

The district court found that "Ancora's use of the word program throughout the '941 Patent only refers to application programs." (A15, underlining added.) This finding is incorrect, as demonstrated below.

The Abstract and the Field of the Invention each reference a "program" broadly, without any mention of an "application."

> **ABSTRACT**
>
> A method of restricting software operation within a license limitation that is applicable for a computer having a first non-volatile memory area, a second non-volatile memory area, and a volatile memory area. The method includes the steps of selecting a program residing in the volatile memory, setting up a verification structure in the non-volatile memories, verifying the program using the structure, and acting on the program according to the verification.

(A42, emphasis added.)

> **FIELD OF THE INVENTION**
>
> This invention relates to a method and system of identifying and restricting an unauthorized software program's operation.

(A45, emphasis added.)

Like the Abstract and the Field of the Invention, the Background of the Invention refers to "programs" without any mention of "applications." (A45.)

The Summary of the Invention begins with a broad reference to "software":

> The present invention relates to a method of restricting <u>software</u> operation within a license limitation. This method strongly relies on the use of a key and of a record, which have been written into the non-volatile memory of a computer.

(A45, 1:38-42, emphasis added.) The next paragraph introduces "a specific non-limiting example" involving an "application," namely Lotus 123. (A45, 1-43 – 2:60-61.) The example concludes with the statement "[t]he example above is given for clarity of explanation only and is by no means binding." (A45, 2:60-61.)

Following this example, the Summary of the Invention expressly describes the invention "in its broadest aspect" with reference to a "program," <u>not</u> an "application."

> In its <u>broadest aspect</u>, the invention provides for a method of restricting software operation within a license limitation including; for a computer having a first non-volatile memory area, a second non-volatile memory area, and a volatile memory area; the steps of: selecting a <u>program</u> residing in the volatile memory, setting up a verification structure in the non-volatile memories, verifying the program using the structure, and acting on the program according to the verification.

(A45, 2:62 – A46, 3:3, emphasis added.)

The Detailed Description uses the term "program" fifteen times, and the term "application" three times. (A47.) Just like the Summary of the Invention, the

Detailed Description expressly introduces an "application" as an "example" of a "program" that may be verified. (A47, 5:28-30.) Nowhere does the Detailed Description state that the term "program" is limited to an "application," or that it somehow excludes an operating system program.

Claim terms enjoy their ordinary meaning unless the specification "clearly" sets forth a special definition. *Elekta Instrument S.A. v. O.U.R. Scientific Int'l, Inc.*, 214 F.3d 1302, 1307 (Fed.Cir.2000); *Cannon Rubber Ltd. v. The First Years, Inc.*, 163 Fed.Appx. 870, 875 (Fed.Cir.2005) ("These two cited instances, however, do not clearly indicate that the patentee intended to assign a more narrow definition to the phrase 'in the body' than it would otherwise possess."). The '941 specification does not "clearly" define (or even attempt to define), the term "program" in a manner that is limited to just "applications" or that would exclude an operating system.

The district court erred when it limited "program" to an express "example" set forth in the specification. *Texas Instruments, Inc. v. United States Int'l Trade Comm'n*, 805 F.2d 1558, 1563 (Fed.Cir.1986) ("This court has cautioned against limiting the claimed invention to preferred embodiments or specific examples in the specification.").

3. The Recitation of "Application Software Program" In Claim 18 Confirms That The Term "Program" In Claim 1 Is Broader Than Just "Software Applications"

The doctrine of claim differentiation indicates that the term "program" in claim 1 is broader than "application." Asserted independent claim 1 broadly recites a "program," whereas unasserted independent claim 18 recites an "application software program." The modification of the term "program" in claim 18 indicates that the unmodified term "program" in claim 1 is broader than "application software." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed.Cir. 2005) (*en banc*).

In *Phillips*, this Court found that the claim term "steel baffles" "strongly implies" that the unmodified term "baffles" is broader, *i.e.*, the term "does not inherently mean objects made of steel." *Id.* In this case, recitation of "application software program" in claim 18 "strongly implies" that the unmodified term "program" in claim 1 is not inherently an "application."

Ultimately, changing the meaning of the term "program" in claim 1 to "application program" found in clam 18 "improperly discounts substantive differences between the claims." *Arlington Industries, Inc. v. Bridgeport Fittings, Inc.*, 632 F.3d 1246, 1254 (Fed. Cir. 2011). Nothing the '941 patent limits the term "program" in claim 1 to "application" programs, or excludes an operating system program.

B. The File History Does Not Narrow The Scope Of "Program"

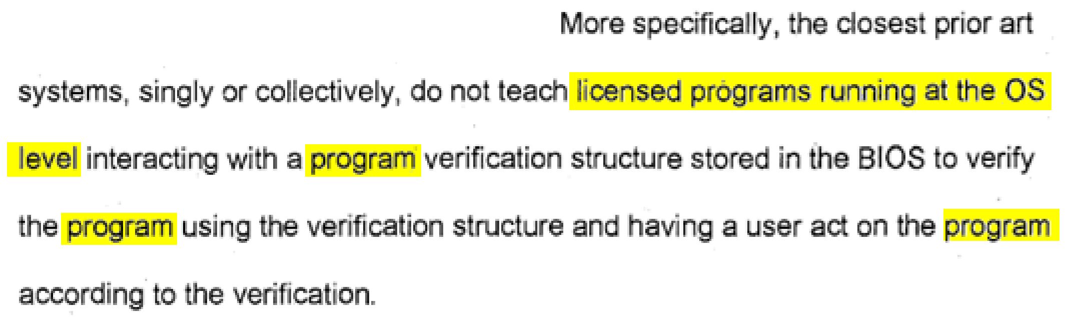
The district court also found that the Applicant disclaimed a broad interpretation of the term "program" during examination. (A17-18.) The district court based its finding on a remark in which the Applicant described his invention as an "application" program. The district court found:

> Moreover, the Patentees specifically described their invention as an application program in order to overcome rejection based upon prior art. This rises to the level of prosecution disclaimer, and clearly, it demonstrates that Patentees and the Patent Examiner understood the invention to apply to application programs. By emphasizing that the '941 Patent is a "[s]oftware license management application[]," Patentees themselves narrowed the ordinary meaning of the claim term.

(A18.)

The Applicant's "software license management application" remark was referring to the "invention" of claim 1, *i.e.*, the software that performs the claimed method steps. (A293.) The statement was <u>not</u> referring to the "program" to be verified – *i.e.,* the "program" on which those method steps operate. The Applicant's remark did not even mention the term "program," let alone exclude operating system programs from the scope of that term in the claim. On the contrary, the Applicant stated that "[s]oftware license management applications, such as the one of the present invention, <u>are operating system (OS) level programs</u>." (A293, emphasis added.)

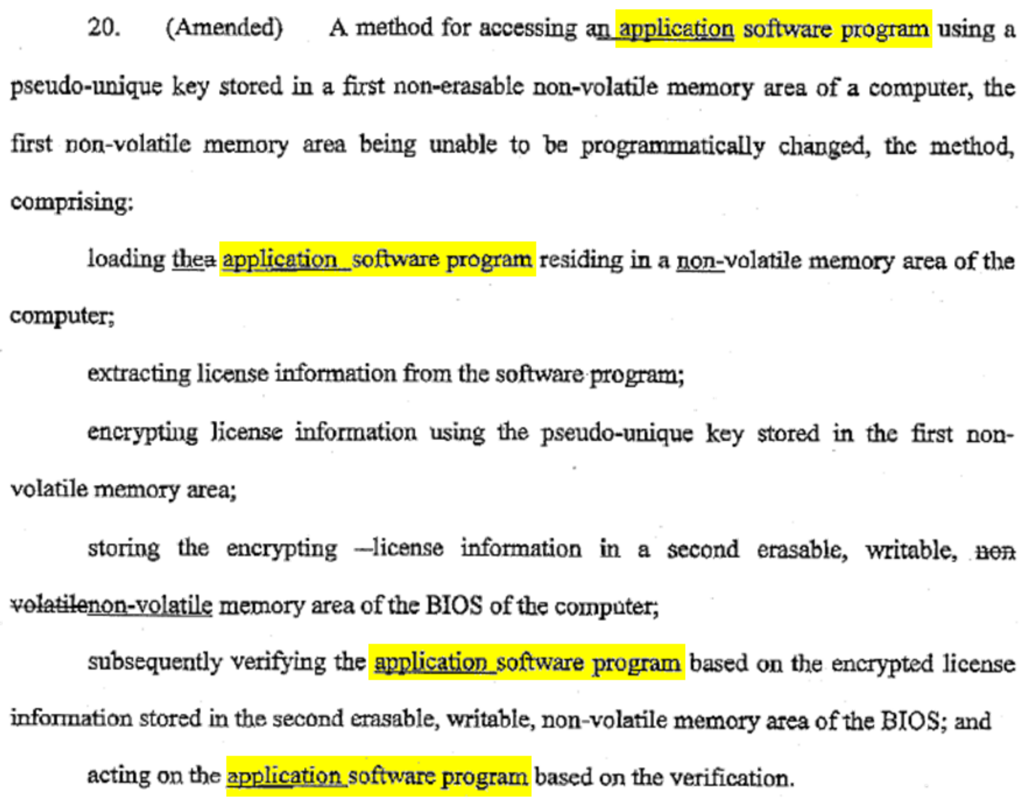
Significantly, in the Reasons for Allowance following these remarks, the Examiner made clear that the "program" to be verified – the "program" at issue on this appeal – is an "OS-level" program:


More specifically, the closest prior art systems, singly or collectively, do not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification.

(A302.)

Prosecution disclaimer requires a "clear and unmistakable" disavowal of claim scope. *Purdue Pharma L.P. v. Endo Pharmaceuticals Inc.*, 438 F.3d 1123, 1136 (Fed. Cir. 2006) (*citing Seachange Int'l, Inc. v. C-COR Inc.*, 413 F.3d 1361, 1372-73 (Fed. Cir. 2005) and *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1323-26 (Fed. Cir. 2003)). Nowhere in the '941 patent or the file history did the Applicant suggest that the claimed "program" to be verified is limited to an "application" program or that it excludes an "operating system" program. On the contrary, the public record on which the public is entitled to rely – e.g. the Examiner's Reasons for Allowance – expressly contradicts the district court's finding that the Applicant had disclaimed operating system software from the scope of the term "program."

Moreover, in the same response the district court cited to find disclaimer with respect to asserted claim 1, the Applicant amended non-asserted claim 18 (claim 20 during examination) to narrow the original claim term "software program" to "application software program." (Underlining added.)



20. (Amended) A method for accessing an application software program using a pseudo-unique key stored in a first non-erasable non-volatile memory area of a computer, the first non-volatile memory area being unable to be programmatically changed, the method, comprising:

loading the~~a~~ application software program residing in a non-volatile memory area of the computer;

extracting license information from the software program;

encrypting license information using the pseudo-unique key stored in the first non-volatile memory area;

storing the encrypting —license information in a second erasable, writable, ~~non volatile~~non-volatile memory area of the BIOS of the computer;

subsequently verifying the application software program based on the encrypted license information stored in the second erasable, writable, non-volatile memory area of the BIOS; and

acting on the application software program based on the verification.

(A297.)

The Applicant did not amend claim 1 in the same fashion. It would be improper to impose a narrowing amendment (amending "software program" to "application software program") from one independent claim onto a claim that was not amended. It would be especially improper here, where the Examiner's Notice of Allowance states that the "programs" to be verified can include "OS level" programs. (A302.) *Arlington Indus., Inc. v. Bridgeport Fittings, Inc.*, 632 F.3d 1246, 1254-55 (Fed. Cir. 2011) (vacating district court's claim construction requiring a "split" configuration where that configuration was added by amendment to other claims, but not the construed claim).

Far from excluding operating system programs from the scope of the term "program," the intrinsic record expressly *includes* them within the scope of claim 1. At the very least, there was no "clear and unmistakable" disavowal of claim scope to support the district court's finding of prosecution disclaimer. *Purdue Pharma*, 438 F.3d at 1136.

C. The District Court's Findings Regarding Operating Systems Are Unsupported And Contrary To The '941 Patent

Following its analysis of the intrinsic record, the district court made certain technical findings regarding operating systems:

> The '941 Patent teaches verification of programs <u>which cannot function unless the operating system is already running</u>. While those programs may then operate at the "same level" as the operating system, a "[s]oftware license management applications, such as the ['941 Patent]," <u>cannot function if the operating system is not functional</u>.

(A18, emphasis added.)

While it is correct that "the '941 Patent teaches verification of programs," nothing in the '941 patent indicates that the claimed verification "cannot function" unless an "operating system is already running." The '941 patent does not even mention an "operating system."

The district court cites no support for its conclusions. They were apparently premised on the incorrect assumption that the claimed "program" is a user application, such as the Lotus 1-2-3 example. As explained above, however, there is no basis for limiting the claims to this example. *Texas Instruments*, 805 F.2d at 1563; *Phillips*, 415 F.3d at 1323; *Arlington Indus.,* 632 F.3d at 1254-55.

To the extent that the district court found that an operating system must be functioning for the verification, and therefore, cannot be the "program" to the verified, such a finding is directly contrary to the '941 patent. The '941 specification states that the method may include "informing the user on the unlicensed status, <u>halting the operation of the program</u>." (A45, 2:24-26, underlining added.) "Halting the operation of the program" inherently means that the program was already running. This embodiment makes clear that the '941 invention may verify a program, such as an operating system, even if program is already running at time of verification.

D. Ancora's Construction Should Be Adopted

For the reasons set forth above, "program" in the '941 patent should be construed according to its ordinary meaning. Apple does not dispute that the ordinary meaning of the term "program" includes an operating system program. (A1474, n. 11.) The district court stated that "to a computer programmer, a program is a 'set of instructions.'" (A14.) Accordingly, Ancora asks this Court to hold that a "program" is "a set of instructions that can be executed by a computer." (A13, A123-125.)

CONCLUSION AND RELIEF SOUGHT

For the foregoing reasons, Ancora asks this Court to reverse the district court's claim construction, order the district court to adopt Ancora's claim construction, vacate the judgment entered pursuant to the erroneous construction, and remand for further proceedings.

Respectfully submitted,

BROOKS KUSHMAN P.C.

/s/ John S. LeRoy
Mark A. Cantor
John S. LeRoy
Marc Lorelli
John P. Rondini
BROOKS KUSHMAN P.C.
1000 Town Center
Twenty-Second Floor
Southfield, Michigan 48075-1238
(248) 358-4400

Attorneys for Plaintiff-Appellant

Date: July 2, 2013

ADDENDUM

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

ANCORA TECHNOLOGIES, INC.,
Plaintiff,

v.

APPLE INC.,
Defendant.

APPLE INC.,
Counterclaim Plaintiff,

v.

ANCORA TECHNOLOGIES, INC.,
Counterclaim Defendant.

Case No. CV 11-06357-YGR

[~~PROPOSED~~] FINAL JUDGMENT

FINAL JUDGMENT

Pursuant to the Court's Claim Construction Order construing claim terms of U.S. Patent No. 6,411,941 (the "'941 patent") and finding that the claims of the '941 patent are not invalid for indefiniteness under 35 U.S.C. § 112, ¶ 2 with respect to the terms "volatile memory" and "non-volatile memory," and the Court's Order Granting Apple Inc.'s Motion for Summary Judgment finding that Apple Inc. has not infringed the '941 patent, the Court ENTERS FINAL JUDGMENT of:

(1) non-infringement of the '941 patent and

(2) non-indefiniteness under 35 U.S.C. § 112, ¶ 2 with respect to the terms "volatile memory" and "non-volatile memory."

The Court also DISMISSES without prejudice Apple's defenses and counterclaims, except for those concerning indefiniteness under § 112, ¶ 2 with respect to the terms "volatile memory" and "non-volatile memory," subject to Apple's right to revive those defenses and counterclaims in the event of a remand.

The parties reserve their rights to challenge any constructions of the disputed claim terms of the '941 patent on appeal.

This is a final, appealable judgment.

IT IS SO ORDERED.

Dated: April 29, 2013

[illegible]
Hon. Yvonne Gonzalez Rogers
United States District Court Judge

United States District Court
Northern District of California

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

ANCORA TECHNOLOGIES, INC., **Plaintiff,** **v.** **APPLE INC.,** **Defendant.**	**Case No.: 11-CV-06357 YGR** **CLAIM CONSTRUCTION ORDER**
AND RELATED COUNTER-CLAIM	

Ancora Technologies, Inc. ("Ancora") alleges that devices that run Apple Inc.'s ("Apple") iOS operating system infringe on U.S. Patent No. 6,411,941 (the " '941 Patent"). Apple has counterclaimed for declaratory judgments of non-infringement and invalidity.

The parties have requested the Court construe seven claim terms/phrases from the '941 Patent: (1) "volatile memory"; (2) "non-volatile memory"; (3) "BIOS"; (4) "program"; (5) "license record"; (6) "verifying the program using at least the verification structure"; and (7) whether the steps in the asserted claims must be performed in a specific order. On June 29, 2012, the parties provided a technology tutorial and on July 11, 2012, the Court held a claim construction hearing.

Based upon the papers submitted, the argument of counsel, for the reasons set forth below, the Court provides the following claim construction.

I. BACKGROUND

The patent in suit relates to software anti-piracy technology. At issue here is technology directed at preventing computer users from copying software and then running that software without a license. Ancora is the owner of the '941 Patent, which claims a method of restricting software operation within a license limitation, *i.e.* it teaches a system for ensuring that only the authorized user of software can operate the software at issue. Apple's iOS operating system also restricts software operation within a license limitation. Ancora alleges that the Apple products that run the iOS operating system infringe on the '941 Patent.

The '941 Patent uses the memory of a computer's "BIOS" to store a "license record" to confirm whether a "program" is licensed to run on that computer. Every computer has a unique identifier embedded at the time of manufacture. Under the teachings of the '941 Patent, when a licensed program first launches it generates a license record using the computer's unique identifier, which license record is stored in the BIOS area of a computer. This license record is unique to that particular computer. When a licensed program is loaded, it can verify whether the software is licensed to run on that computer by referencing the license record stored in the BIOS with the license record from the program. If they match, the program continues to run. If the program has been copied, the license information does not match and the program will not run.

A. Background of the Patent

Plaintiffs provide the following background: In 1997, when Miki Mullor and Julian Valiko, the co-inventors of the '941 Patent ("Patentees"), began developing the technology that would become the '941 Patent, there were two approaches to combating software piracy, a hardware approach and a software approach. The hardware approach was costly, inconvenient and not suitable for software downloaded from the internet required as it required users of software to use a piece of hardware called a "dongel" in order to access the software. The software based products were too easily hacked by skilled programmers.

Patentees developed a third approach that had the advantages of both the hardware approach and software approach without the disadvantages of either. Patentees identified available memory space in hardware stored on the computer's motherboard, the BIOS, which they repurposed to store software licensing technology. The inventive aspect of the '941 Patent is that the writable, non-volatile memory of the BIOS is not ordinarily considered to be a storage medium for software licensing technology. The advantage of using the BIOS for this purpose is that the level of programming expertise required to tamper with data stored in the BIOS is substantially greater than the expertise needed to tamper with data residing in volatile memory, and unsuccessful tampering comes with higher risk as it could render the computer inoperable.

Patentees applied for an Israeli patent in 1998. On October 1, 1998, Patentees applied for the ’941 Patent, with a priority date of May 21, 1998 based upon the Israeli patent. The ’941 Patent issued in 2002.

B. Claim Terms/Phrases to be Constructed

Sixteen claims from the ’941 Patent are asserted: independent Claim 1, and dependent Claims 2, 3, and 5-17, which refer to it.

Claim 1, which is the only independent claim asserted, recites the following (the language the parties have identified for construction is in bold and italics):

> 1. A method of restricting software operation within a license for use with a computer including an erasable, ***non-volatile memory*** area of a ***BIOS*** of the computer, and a ***volatile memory*** area; the method comprising the steps of:
>
> selecting a ***program*** residing in the ***volatile memory***, using an agent to set up a verification structure in the erasable, ***non-volatile memory*** of the ***BIOS***, the verification structure accommodating data that includes at least one ***license record***,
>
> ***verifying the program using at least the verification structure*** from the erasable ***non-volatile memory*** of the ***BIOS***, and acting on the ***program*** according to the verification.

(’941 Patent, claim 1).

The parties request the Court construe seven claim terms/phrases: (1) “volatile memory”; (2) “non-volatile memory”; (3) “BIOS”; (4) “program”; (5) “license record”; (6) “verifying the program using at least the verification structure”; and (7) All Asserted Claims.[1]

II. PRINCIPLES OF CLAIM CONSTRUCTION

Claim construction is a matter of law, to be decided by the Court. *Markman v. Westview Instruments, Inc.*, 517 U.S. 370, 387 (1996) (determination of infringement is a two-step analysis: First, the Court determines the scope and meaning of the claims; second, the properly construed claims are compared to the accused device.). “[T]he role of a district court in construing claims is … to give meaning to the limitations actually contained in the claims, informed by the written

[1] In addition, the parties have identified one term on which they have agreed on a construction (“verification structure accommodating data that includes at least one license record”).

description, the prosecution history if in evidence, and any relevant extrinsic evidence." *American Piledriving Equipment, Inc. v. Geoquip, Inc.*, 637 F.3d 1324, 1331 (Fed. Cir. 2011). "Claim construction is a matter of resolution of disputed meanings and technical scope, to clarify and when necessary to explain what the patentee covered by the claims, for use in the determination of infringement." *U.S. Surgical Corp. v. Ethicon, Inc.*, 103 F.3d 1554, 1568 (Fed. Cir. 1997). Thus, claim terms need only be construed "to the extent necessary to resolve the controversy." *Wellman, Inc. v. Eastman Chemical Co.*, 642 F.3d 1355, 1361 (Fed. Cir. 2011) (citing *Vivid Technologies, Inc. v. American Science & Engineering, Inc.*, 200 F.3d 795, 803 (Fed. Cir. 1999).[2]

The starting point in a claims construction analysis is the language of the claims themselves. These define the invention that the patentee may exclude others from practicing. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2005). The general rule is to construe a claim term in a manner consistent with its "ordinary and customary meaning," which is "the meaning that the term would have to a person of ordinary skill in the art in question at the time of the invention." *Id.* at 1312.

Claims must be read in view of the specification, of which they are a part and in a manner consistent with the patent's specification. *See Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 979 (Fed. Cir. 1995), *aff'd*, 517 U.S. 370 (1996). The specification may act as a sort of dictionary, which explains the invention and may define terms used in the claims. *Id.* The Court also should consider the patent's prosecution history, if it is in evidence. *Id.* at 980. The prosecution history may "inform the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be." *Phillips*, *supra*, 415 F.3d at 1317 (citing *Vitronics*, 90 F.3d at 1582-83); *see also Chimie v. PPG Indus., Inc.*, 402 F.3d 1371, 1384 (Fed. Cir.

---

[2] Once the meaning of a term used in a claim has been determined, the same meaning applies to that term for all claims in which the same term appears. *Inverness Med. Switzerland GmbH v. Princeton Biomeditech Corp.*, 309 F.3d 1365, 1371 (Fed. Cir. 2002). After a term is construed, the Court's construction becomes the legally operative meaning of the disputed terms that governs further proceedings in the case. *See Chimie v. PPG Indus., Inc.*, 402 F.3d 1371, 1377 (Fed. Cir. 2005). However, "district courts may engage in a rolling claim construction, in which the court revisits and alters its interpretation of the claim terms as its understanding of the technology evolves." *Pressure Products Medical Supplies, Inc. v. Greatbatch Ltd.*, 599 F.3d 1308, 1316 (Fed. Cir. 2010).

2005) ("The purpose of consulting the prosecution history in construing a claim is to exclude any interpretation that was disclaimed during prosecution.") (internal quotations omitted). The Court may, in its discretion, consider extrinsic evidence[3] if such sources will aid the Court in determining "the true meaning of language used in the patent claims." *Phillips*, *supra*, 415 F.3d at 1318.

Further, and as relevant here, whether a patent claim complies with the definiteness requirement of 35 U.S.C. § 112, ¶ 2 is also a matter of claim construction. *See Wellman, Inc. v. Eastman Chemical Co.*, 642 F.3d 1355, 1365-66 (Fed. Cir. 2011), *cert. denied*, 132 S.Ct. 1541 (2012). Section 112, paragraph 2 of the Patent Act provides in pertinent part: "[t]he specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention." 35 U.S.C. § 112, ¶ 2. This section contains two requirements: "first, [the claim] must set forth what 'the applicant regards as his invention,' and second, it must do so with sufficient particularity and distinctness, *i.e.*, the claim must be sufficiently 'definite.'" *Allen Eng'g Corp. v. Bartell Indus., Inc.*, 299 F.3d 1336, 1348 (Fed. Cir. 2002); *see also*, *Phillips*, *supra*, 415 F.3d at 1316. In determining whether a claim is sufficiently definite, the Court must consider whether "one skilled in the art would understand the bounds of the claim when read in light of the specification." *Allen Eng'g Corp.*, *supra*, 299 F.3d at 1348 (citing *Personalized Media Comm'ns, LLC v. Int'l Trade Comm'n*, 161 F.3d 696, 705 (Fed. Cir. 1998). "Only claims 'not amenable to construction' or 'insolubly ambiguous' are indefinite." *Halliburton Energy Services, Inc. v. M-I LLC*, 514 F.3d 1244, 1250 (Fed. Cir. 2008) (quoting *Datamize, LLC v. Plumtree Software, Inc.*, 417 F.3d 1342, 1347 (Fed. Cir. 2005).

III. DISCUSSION

A. THE FIRST AND SECOND DISPUTED CLAIM TERMS – "VOLATILE MEMORY" & "NON-VOLATILE MEMORY" (CLAIMS 1-3, 5-17)

The parties' dispute focuses on the impact of whether examples provided in the specification render the claim indefinite.

[3] Although the use of extrinsic evidence is discretionary, the court may always consult technical treatises and dictionaries to understand the technology and to construe the claims, so long as no definition in the intrinsic evidence is contradicted.

The parties' proposed constructions are shown below:

TERM	APPLE'S PROPOSED CONSTRUCTION	ANCORA'S PROPOSED CONSTRUCTION
"volatile memory"	This term is indefinite under 35 U.S.C. § 112, ¶ 2.	memory that is not maintained when the power is removed
"non-volatile memory"	This term is indefinite under 35 U.S.C. § 112, ¶ 2.	memory that is maintained when the power is removed

Claim 1 calls out two different types of memory—volatile and non-volatile. The plain and ordinary meaning of the terms is not in dispute. To one of ordinary skill in the art, a volatile memory is memory whose data is not maintained when the power is removed and a non-volatile memory is memory whose data is maintained when the power is removed.

However, the '941 Patent specifically indicates that "volatile memory" can take the form of a hard disk. Apple takes issue with this pronouncement. A "hard disk" is generally known by those of ordinary skill in the art to be *non*-volatile memory because the memory is maintained when the power is removed. Apple argues that the Patentees' express identification of a hard disk as "volatile memory" runs counter to the understanding of one of ordinary skill in the art, and therefore, renders the terms indefinite under 35 U.S.C. § 112, ¶ 2. Next, it argues that a definition of "volatile memory" as "memory that is not maintained when the power is removed" excludes the very same hard disk that the specification identifies as volatile. Furthermore, Apple argues that this ambiguity affects the meaning of "non-volatile memory" by implication, which makes it impossible to determine the scope of both terms as those terms are used in the '941 Patent. According to Apple, this creates an insoluble ambiguity as to what characteristics render memory volatile or non-volatile. Thus, Apple argues that although the specification identifies only "hard disk" as a form of volatile memory, this insoluble ambiguity applies to other types of memories as well because there is no explanation of what characteristics render memory volatile or non-volatile. According to Apple, "[i]t is difficult to imagine a more compelling example of a situation in which '[c]ompetitors trying to practice the invention or to design around it would be unable to discern the bounds of the invention.'" (Opp'n 9 (quoting *Honeywell Int'l, Inc. v ITC*, 341 F.3d 1332, 1341 (Fed. Cir. 2008).)

Ancora argues that the claim terms should not be constructed at all because the plain and ordinary meaning of the terms is not in dispute and the structure disclosed in the specification is irrelevant to the claim construction analysis.[4] Ancora urges the Court to perform a claim construction divorced from the context of the specification and prosecution history. *Old Town Canoe Co. v. Confluence Holdings Corp.*, 448 F.3d 1309, 1318 (Fed. Cir. 2006).

Claims are not indefinite merely because they present a difficult task of claim construction. *See Liebel-Flarsheim Co. v. Medrad, Inc.*, 358 F.3d 898, 911 (Fed. Cir. 2004) ("unless the court concludes, after applying all the available tools of claim construction, that the claim is still ambiguous, the axiom regarding the construction to preserve the validity of the claim does not apply."). Instead, "[i]f the meaning of the claim is discernible, even though the task may be formidable and the conclusion may be one over which reasonable persons will disagree, we have held the claim sufficiently clear to avoid invalidity on indefiniteness grounds." *Halliburton Energy Services, Inc. v. M-I LLC*, 514 F.3d 1244, 1249 (Fed. Cir. 2008) (citing *Exxon Research & Eng'g Co. v. United States*, 265 F.3d 1371, 1375 (Fed. Cir. 2001) (citations omitted).

Ancora explains that volatile memory is routinely stored on the "hard disk" when two programs attempt to write data to the same location in RAM (random access memory), which it refers to as "virtual data." Thus, Ancora argues that it is entirely appropriate to consider a "hard disk" as a form of volatile memory to someone of ordinary skill in the art. Apple disagrees. Apple's view may be appealing to a layman; the same may not be true for someone of ordinary skill in the art.

Based on the foregoing analysis, the Court declines to find the term indefinite as a matter of law and finds that the claim terms should be given their ordinary meaning, specifically:

Volatile memory is memory whose data is not maintained when the power is removed; and non-volatile memory is memory whose data is maintained when the power is removed.

[4] Ancora also argues that construction of this term is unnecessary for the infringement analysis because the term hard disk does not appear in Claim 1 and Apple's accused devices do not store the license record in a hard-drive. However, Apple has raised a counterclaim of invalidity, and indefiniteness is a basis to find a patent invalid.

B. Third Disputed Claim Term/ Phrase – "BIOS" (Claims 1-3, 5-17)

No party disputes that "BIOS" is an acronym for Basic Input/Output System. Rather, the parties dispute whether the term "BIOS" applies to all computers or only to IBM PC-compatible computers, as Apple proposes.[5] The parties' proposed constructions are shown below:

Apple's Proposed Construction	Ancora's Proposed Construction
software routines on IBM PC compatible computers that handle startup operations and support the transfer of data among peripheral devices	software routines that handle startup operations

The claims of the '941 Patent refers to the "non-volatile memory area of the BIOS" of a computer fourteen times (and refers to the "non-volatile memory area of *a* BIOS" once). Neither the claim nor the specification defines "BIOS." In the prosecution history the Patent Examiner gave the following definition of BIOS based upon "The Microsoft Computer Dictionary, 5th Edition, 2002":

> the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices.

The Patent Examiner further commented:

> This definition is consistent with the specification of the '941 patent. Since a BIOS is therefore defined by the functional descriptive material contained within it, one skilled in the art would consider any non-functional descriptive material, such as tables, to be part of the BIOS only if it is made and used by the functions of the BIOS itself. This does not preclude such material being also used or modified by programs located outside of the BIOS, such as applications running in an operative system. The fact that a program or tables resides in non-volatile memory does not necessarily mean that it is part of the BIOS. It is therefore the case that a reasonable examiner would only consider a table to be in a BIOS if it were, at a minimum, created by a function residing in the BIOS.

(Rondini Dec. ¶ 4, Ex. 3, Order Granting Request for Reexamination, at 8-9, ANCA 2568-69.)

[5] Apple also seeks to limit the scope of the definition so that BIOS handles transfer of data among "peripheral" devices. Ancora argues that the Claim does not mention "peripheral" devices and that the Examiner used the broader "hardware" language not the narrower term "peripheral." In a footnote, Apple states that its construction is consistent with the definitions in the technical dictionaries, but Apple's proposed construction is consistent with only one of the five the technical dictionaries quoted in Apple's *Markman* Brief. Apple has not otherwise explained why it believes that BIOS should be construed to interact with peripheral devices only. The Court has not found any basis to support this interpretation.

Apple points out that the definition supplied by the Patent Examiner is incomplete, referring again to the Microsoft dictionary:

> BIOS n. Acronym for basic input/output system. On PC-compatible computers, the set of essential software routines that tests hardware at startup, starts the operating system, and supports the transfer of data among hardware devices, including the date and time. The operating system date is initialized from the BIOS or Real Time Clock date when the machine is booted. Many older PCs, particularly those dating before 1997, have BIOSs that store only 2-digit years and thus may have suffered from Year 2000 problems. The BIOS is stored in read-only memory (ROM) so that it can be executed when the computer is turned on. Although critical to performance, the BIOS is usually invisible to computer users.

(*Id*. Ex. I (Microsoft Press Computer User's Dictionary (5th ed. 2002)).)[6]

[6] In addition to the dictionary definition used by the Patent Examiner, Apple offers four other dictionary definitions of the term BIOS:

> BIOS n. Acronym for basic input/output system. On PC-compatible computers, the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices. The BIOS is stored in ROM so that it can be executed when the computer is turned on. Although critical to performance, the BIOS is usually invisible to computer users.

(Rahebi Dec., Ex. K (Microsoft Press Computer User's Dictionary (3d ed. 1998)).)

> A set of programs encoded in read-only memory (ROM) on IBM PC-compatible computers. These programs handle startup operations such as the power-on self-test (POST) and low-level control for hardware, such as disk drives, keyboards, and monitor. The BIOS programs of IBM personal computers are copyrighted, so manufacturers of IBM PC-compatible computers must create BIOSs that emulate the IBM BIOS or buy an emulation from companies, such as Phoenix Technologies and American Megatrends, Inc. ….

(*Id*. Ex. F (Webster's New World Dictionary of Computer Terms (6th ed. 1997)); *id*. Ex. G (Que's Computer & Internet Dictionary (6th ed. 1995)).)

> [A] set of procedures stored on a ROM chip inside IBM PC compatible computers. These routines handle all input-output functions, including screen graphics, so that programs do not have to manipulate the hardware directly ….

(*Id*. Ex. H (Barron's Dictionary of Computer and Internet Terms (5th ed. 1996)).)

> On PC-compatible computers, the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices. The BIOS is stored in read-only memory (ROM) so that it can be executed when the computer is turned on. Although critical to performance, the BIOS is usually invisible to computer users.

(*Id*. Ex. E (Microsoft Press Computer Dictionary (3d ed. 1997)).)

> The part of the system software of the IBM PC and compatibles that provides the lowest level interface to peripheral devices and controls the first stage of the bootstrap process, including installing the operating system. The BIOS is stored in ROM, or equivalent, in every PC. Its main task is to load and execute the operating system which is usually stored on the computer's hard disk, but may be loaded from CD-ROM or floppy disk at install time.

(*Id*. Ex. J (Free On-Line Dictionary of Computing (June 6, 1999)).)

Apple proposes that BIOS be construed to operate only on IBM PC-compatible computers. According to Apple, the plain and ordinary meaning of the term "BIOS" at the time of the invention, as demonstrated by dictionary definitions is that BIOS is specific to the IBM PC-compatible computer platform.

Ancora urges the Court to avoid using a dictionary to construe BIOS by arguing that the Federal Circuit in *Phillips* stated that using a dictionary to alter the claim term violates the public notice function of the patent. The problem identified in *Phillips* is that a dictionary definition oftentimes will be overly broad and when taken out of the context of the patent at issue can lead to an "unduly expansive" construction of the claim term. 415 F.3d at 1321. The opposite problem is presented here. Apple is resorting to the dictionary definition to narrow the construction of the term because it believes that the Patent Examiner gave BIOS an unduly expansive construction.

Ancora argues that the Examiner defined BIOS broadly, not limited to IBM PC-compatible computers, and that the claim itself is not limited to a brand of computer but the "BIOS of a computer."[7] Ancora argues that virtually all computers have BIOS, including the accused devices (iPhone, Apple laptops and Apple desktops).[8]

As set forth above, "BIOS" stands for <u>B</u>asic <u>I</u>nput/<u>O</u>utput <u>S</u>ystem; it is software code. No one disputes that a person of ordinary skill in the art reading the Claim in the context of the specification and prosecution history would understand the "BIOS" to be the location in the computer where the software code was stored. The inventive aspect of the '941 Patent was to write information onto unused memory in the BIOS area of the computer. The limiting aspect of the invention is to store information in the BIOS, not the type of computer that runs BIOS. The technical dictionaries that Apple has offered into evidence do not convince the Court otherwise.

[7] Although Ancora argues that the Patent Examiner gave BIOS a specific definition, "the set of essential software routines that test hardware at startup, start the operating system, and support the transfer of data among hardware devices," Ancora argues for an entirely different construction of BIOS tethered only to Apple's proposed construction. Other than to argue that its patent applies to non-IBM PC-compatible computers, Ancora does not base its interpretation on any intrinsic evidence.

[8] Ancora also offers evidence of a 1983 Commodore 64 computer that ran BIOS, and that during the mid-1990's Apple's Macintosh brand of computers, when operated in a PC-compatible mode, had BIOS. This extrinsic evidence both before and after the relevant time period does not reflect the usage or meaning in the art in 1998.

These dictionaries indicate that PC-compatible computers run BIOS. However, exclusionary language is not found in the proffered dictionaries, *i.e.* the dictionaries do not indicate that the non-IBM PC-compatible computers do not or cannot run BIOS, nor that a person of ordinary skill in the art would have understood as much. Accordingly, the Court will not define BIOS by the hardware architecture of the computer on which it runs.

The Court rejects both parties' proffers and provides the following construction:

"BIOS" is an acronym for Basic Input/Output System. It is the set of essential startup operations that run when a computer is turned on, which tests hardware, starts the operating system, and supports the transfer of data among hardware devices.

C. FOURTH DISPUTED CLAIM TERM/ PHRASE – "PROGRAM" (CLAIMS 1-3, 5-17)

The parties dispute whether an "operating system" is a type of "program" as that term is used in the '941 Patent. The parties' proposed constructions are shown below:

APPLE'S PROPOSED CONSTRUCTION	ANCORA'S PROPOSED CONSTRUCTION
Software application that interacts with and relies on the operating system	A set of instructions that can be executed by a computer.

Apple argues that (i) using the term program in the specification to refer only to application programs and then (ii) emphasizing the distinction during prosecution demonstrates a narrow use of the term. Ancora argues that (i) the specification does not clearly set forth a narrower definition of the term; (ii) the prior art reference over which the Examiner allowed the '941 Patent to issue described an operating system as a type of program; and (iii) there is no disavowal of the plain and ordinary meaning of the term in the prosecution history.

According to Ancora, "[e]very person of skill in the computer field knows that a 'program' is: 'a set of instructions that can be executed by a computer.'" (Ancora's Br. 12.) The construction that Ancora advances is the definition in the Microsoft Computer Dictionary, 5th Edition, 2002. By contrast, Apple argues that to a person of skill in the field, the term "program" as used in the '941 Patent means a "software application that interacts with and relies on the operating system." As explained below, Ancora's definition is too broad, while Apple's may be misunderstood by a jury.

The first step is to look at the language of the use of the term in the claim itself: the ’941 Patent teaches a method of “… selecting a program residing in the volatile memory, … verifying the program …, and acting on the program.”

Ancora argues that no construction is necessary because the term “program” is a commonly understood term. However, the term program has a different meaning depending on the context. *See REC Software USA, Inc. v. Bamboo Solutions Corp.*, 2012 WL 1533965, at *5 (W.D. Wash. April 30, 2012) (“the word has numerous meanings, *e.g.*, a computer program versus a baseball game program versus a television program.”). A person attending a play would consider a program to be the list of the performers or performances; a person watching a television show would consider the show itself to be the program; to a computer programmer, a program is a “set of instructions.” *See id.* The ’941 Patent is this latter type of program. Due to the technical nature of the term, the Court believes that construction is necessary to resolve the dispute over the meaning of the term. *See U.S. Surgical Corp.*, *supra*, 103 F.3d at 1568 (claim construction may be necessary to resolve disputes over the meaning or scope of technical terms and words of art). The issue then is whether the invention applies to all types of computer programs, as Ancora argues, or application programs only, as Apple argues.

1. Specification

A technical term in a patent claim is construed in accordance with its description in the specification. *Phillips, supra*, 415 F.3d at 1315. Apple argues that the summary of the invention refers to “application program” (’941 Patent 1:53-54), and the specification refers to a “license verifier application” (*id.* 2:13-14), which can only run if the operating system is already running and thus could not verify the operating system. Apple also notes that the only “program” expressly identified in the specification is “Lotus 123,” an application program. Ancora counters that the term “program” is used broadly within the specification to include “software” and “application,” which demonstrates the breadth of the term rather than a limitation on the definition.

Words of a claim are to be given the meaning they would have to a person of ordinary skill in the art, who read the claims in the context of the description of the invention in the specification and the prosecution history. *See Phillips, supra*, 415 F.3d at 1313. The ordinary and customary meaning of a claim term is not the dictionary definition[9] as Ancora would like, but the meaning of that claim term to a person of ordinary skill in the art "after reading the entire patent." *Id.* at 1320 ("Properly viewed, the 'ordinary meaning' of a claim term is its meaning to the ordinary artisan after reading the entire patent."). In reviewing the intrinsic record, this Court must also "strive to capture the scope of the actual invention, rather than strictly limit[ing] the scope of the claims to disclosed embodiments or allow[ing] the claim language to become divorced from what the specification conveys is the invention." *Retractable Techs., Inc. v. Becton*, 653 F.3d 1296, 1305 (Fed. Cir. 2011). There is a fine line between construing claims in light of their specification and improperly importing a limitation into a claim. *See Phillips*, *supra*, 415 F.3d at 1323. The distinction between proper claim construction and improper limitation turns on "whether a person of skill in the art would understand the embodiments to define the outer limits of the claim term or merely to be exemplary in nature." *Id*.

Ancora's use of the word program throughout the '941 Patent only refers to application programs. The '941 Patent teaches only a method of restricting the operation of unauthorized application programs. While the '941 Patent does not repeatedly use of the phrase "application program," it does not need the modifier for it to be clear that the '941 Patent does not teach a method of restricting the operation of all types of programs. From the outset of the specification of the '941 Patent, Ancora limits the nature of a "program." Thus:

> FIELD OF THE INVENTION
>
> This invention relates to a method and system of identifying and restricting an unauthorized software program's operation.

---

[9] The Federal Circuit in *Philips* instructed courts to avoid an over-expansive definition of claim terms: courts should "focus[ ] at the outset on how the patentee used the claim term in the claims, specification, and prosecution history, rather than starting with a broad definition and whittling it down." *Philips, supra*, at 1321. The *Phillips* Court's concern with dictionaries is that it may lead judges to construe terms in an overbroad manner:

> if the district court starts with the broad dictionary definition in every case and fails to fully appreciate how the specification implicitly limits that definition, the error will systematically cause the construction of the claim to be unduly expansive.

Id.

BACKGROUND OF THE INVENTION

Numerous methods have been devised for the identifying and restricting of an unauthorized software program's operation.

('941 Patent 1:5-14.)

As described in the '941 Patent specification, the Patentees' purpose focused solely on the prevention of "illegal copying" of software. Ancora's use of the word "application" to modify the word "program" emphasizes the nature of the program at issue. While at times redundant, the language does not undermine the fundamental teachings of the patent. To assert that the patent teaches anything with respect to non-application based programs, such as an operating system, finds no basis in the language of the patent itself.

Based on the foregoing analysis, the Court concludes that the specification does not support Ancora's position that "program" is defined broadly in the '941 Patent.

2. *Prosecution history*

The prosecution history "provides evidence of how the PTO and the inventor understood the patent" and "can often inform the meaning of the claim language by demonstrating how the inventor understood the invention and whether the inventor limited the invention in the course of prosecution, making the claim scope narrower than it would otherwise be." *Phillips*, *supra*, 415 F.3d at 1317 (citing *Vitronics*, *supra*, 90 F.3d at 1582-83).

Apple argues that the prosecution history confirms that the claims are directed to confirming the license of programs running at the operating system level and not confirming the operating system itself, because, according to Apple, Ancora disavowed a broad definition of program in the patent prosecution.

"Under the doctrine of prosecution disclaimer, a patentee may limit the meaning of a claim term by making a clear and unmistakable disavowal of scope during prosecution." *Purdue Pharma L.P. v. Endo Pharmaceuticals Inc.*, 438 F.3d 1123, 1136 (Fed. Cir. 2006) (citing *Seachange Int'l, Inc. v. C-COR Inc.*, 413 F.3d 1361, 1372-73 (Fed. Cir. 2005); *Omega Eng'g, Inc. v. Raytek Corp.*, 334 F.3d 1314, 1323-26 (Fed. Cir. 2003)). A patentee may disavow the scope of the meaning of a claim term by, for example, clearly characterizing the invention in a way to try to overcome rejections based on prior art. *See, e.g., Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1349 (Fed. Cir. 2004) (limiting the term "transmitting" to require direct transmission over telephone line because the patentee stated during prosecution that the invention transmits over a standard telephone line, thus disclaiming transmission over a packet-switched network); *Alloc v. Int'l Trade Comm'n*, 342 F.3d 1361, 1372 (Fed. Cir. 2003) (finding the patentee expressly disavowed floor paneling systems without "play" because the applicant cited the feature during prosecution to overcome prior art); *Bell Atl. Network Servs. v. Covad Commc'ns Group, Inc.*, 262 F.3d 1258, 1273 (Fed. Cir. 2001) (limiting operation of the "transceiver" to the three stated modes because of clearly limiting statements made by the patentee to try to overcome a prior art rejection). Where the patentee has unequivocally disavowed a certain meaning to obtain the patent, the doctrine of prosecution disclaimer attaches and narrows the ordinary meaning of the claim congruent with the scope of the surrender. *Cordis Corp. v. Medtronic Ave, Inc.*, 511 F.3d 1157, 1177 *supplemented sub nom. Cordis Corp. v. Boston Scientific Corp.*, 275 F. App'x 966 (Fed. Cir. 2008) ("an applicant can make a binding disavowal of claim scope in the course of prosecuting the patent, through arguments made to distinguish prior art references.") (citing *Conoco, Inc. v. Energy & Envtl. Int'l, L.C.*, 460 F.3d 1349, 1364 (Fed. Cir. 2006); *Pharmacia & Upjohn Co. v. Mylan Pharms., Inc.*, 170 F.3d 1373, 1376 (Fed. Cir. 1999); *Litton Sys., Inc.*, 140 F.3d at 1458).

Here, the Patent Examiner rejected all Claims, including Claim 1, as being unpatentable as obvious based on the prior art, Misra *et al.*, U.S. Patent No. 6,189,146, Goldman *et al.*, U.S. Patent No. 5,684,951, and Ewertz *et al.*, U.S. Patent No. 5,479,639. Patentees distinguished that prior art by stating that Misra *et al.* teaches a method for enforcing software licenses that describes the step of generating a license IDs for, *inter alia*, an Operating System program but fails to teach using the BIOS of a computer to store that license ID. In distinguishing that prior art, Patentees emphasized that "[s]oftware license management applications, such as the one of the present invention [the '941 Patent], are operating system (OS) level programs," which "do not ordinarily interact or communicate" with the BIOS. (Ex. 2, ANCA 713.) Subsequently the Patent Examiner allowed the Claims and emphasized that the prior art "do not teach licensed programs running at the OS level interacting with a program verification structure stored in the BIOS to verify the program using the verification structure and having a user act on the program according to the verification." (*Id.*, ANCA 722.) Both parties believe that this supports their construction.

Statements in the prosecution history, similar to the specification, describe the invention as an application program. This demonstrates what Patentees considered to be their invention and consequently, what the Patent Examiner would have understood Patentees' invention to be. Moreover, the Patentees specifically described their invention as an application program in order to overcome rejection based upon prior art. This rises to the level of prosecution disclaimer, and clearly, it demonstrates that Patentees and the Patent Examiner understood the invention to apply to application programs. By emphasizing that the '941 Patent is a "[s]oftware license management application[ ]," Patentees themselves narrowed the ordinary meaning of the claim term.

Further, Ancora's argument makes no logical sense; nor does the record support the broad construction it proposes. The '941 Patent teaches verification of programs which cannot function unless the operating system is already running. While those programs may then operate at the "same level" as the operating system, a "[s]oftware license management applications, such as the ['941 Patent]," cannot function if the operating system is not functional.

Based on the foregoing analysis, the Court finds that the term "program" as used in the '941 Patent means:

A set of instructions for software applications that can be executed by a computer.

D. Fifth Disputed Claim Term/ Phrase – "license record" (Claims 1-3, 5-17)

The parties disagree as to whether the term "license record" is a record that identifies the licensed program and the number of licensed users, as Apple urges, or more broadly, information for verifying a licensed program, as Ancora contends. The parties' proposed constructions are shown below:

Apple's Proposed Construction	Ancora's Proposed Construction
record from the licensed program that identifies the licensed program and the number of licensed users	information for verifying a licensed program

The ’941 Patent stores a “license record” in the BIOS area of a computer. Neither the claim nor the specification defines “license record.”

Apple argues that a license record must provide data about the nature of the license for it to be a record, and therefore, the license record “identifies the licensed program and the number of licensed users.” (’941 Patent 6:9-16.) Apple bases its construction on two non-limiting examples provided in the specification. The first non-limiting example identifies the following information as part of a license record: “author name, program name, and number of licensed users (for network).” (*Id.* 1:55-57.) Additionally, the “Detailed Description of a Preferred Embodiment” provides as an example of what the license record includes: “Application names [*sic*] (e.g. Lotus 123), Vendor name (Lotus inc. [*sic*]), and number of licensed copies (1 for stand alone [*sic*] usage, >1 for number of licensed users for a network application).” (*Id.* 5:29-33.) Ancora argues that these non-limiting “examples” cannot be used to limit the claim term.

The non-limiting examples provide examples of information that a license record *may* contain, but not what information a license record *must* contain. In the specification’s first example, “the number of licensed users” is information included in the license record “for network” computers, which suggests that the number of licensed users is not a required part of the license record for non-network computers. Moreover, the specification provides that the contents of a license record “may include terms, identifications, specifications, or limitations related to the manufacturer of a software product, the distributor of a software product, the purchaser of a software product, a licensor, a licensee, items of computer hardware or components thereof, or to other terms and conditions related to the aforesaid.” (*Id.* 6:11-17.) Nothing in the non-limiting examples contained in the specification requires that a license record identify the number of licensed users, as Apple urges.

Apple argues that Ancora’s proposed construction would render claim language meaningless because Ancora’s construction fails to distinguish between “license record” and “license information” by construing license record to mean *any* information verifying a licensed program without any requirement that the information indicate anything about the nature of the license. In so doing, Apple attempts to use dependent claim 15, as well as independent claim 18 and claim 19, which is dependent on claim 18 (all three referring to "license information") to limit independent claim 1 (referring to "license record"). While Apple is correct that the licensed record must contain information about the license for it to be a record, the specification does not support Apple's conclusion regarding the specific nature of the contents of a license record.

Based on the foregoing analysis, the Court concludes that Apple's construction is too narrow, while Ancora's construction is too broad. The Court provides the following definition for "license record":

A record from a licensed program with information for verifying that licensed program.

E. SIXTH DISPUTED CLAIM TERM/ PHRASE – "VERIFYING THE PROGRAM USING AT LEAST THE VERIFICATION STRUCTURE" (CLAIMS 1-3, 5-17)

The parties dispute whether verifying that a program is licensed requires a comparison between the license record extracted from the program to the license record in the verification structure, or whether it can cover any method of verification, using at least the verification structure. The parties' proposed constructions are shown below:

APPLE'S PROPOSED CONSTRUCTION	ANCORA'S PROPOSED CONSTRUCTION
confirming whether the program is licensed by comparing the license record extracted from the program to the license record in the verification structure	confirming whether a program is licensed using at least the verification structure

This phrase, which appears in Claim 1 of the '941 Patent is not defined in either the Claim or the specification.

For simplicity, the Court will divide this Claim phrase, and the parties' constructions thereof, into three component parts: (1) "verifying the program"; (2) "using at least"; and (3) "the verification structure." The parties' proposed constructions of the first and third parts of the Claim phrase are virtually identical. Apple and Ancora's propose construing "verifying the program" as "confirming whether *the* program is licensed" and "confirming whether *a* program is licensed," respectively. Both parties agree that "the verification structure" means "the verification structure." Thus, the dispute really is over the meaning of "using at least." That is, whether "using at least" means "using at least," as Ancora urges, or whether "using at least" means "by comparing the license record extracted from the program to the license record in," as Apple urges.

Ancora argues that a comparison between the license record extracted from the program and the license record in the verification structure is not required, while Apple argues verification is not possible without such a comparison. Apple argues that the specification describes the verification process, which includes a comparison between the license record extracted from the program and the license record in the verification structure. Ancora notes that the specification describes multiple examples of techniques to verify a program, including verification by determining whether a particular program is "compatible" with the license record. ('941 Patent, 3:38-41.) According to Ancora, "comparison" is not required for this compatibility verification.

The Court cannot find any limitation in the Claim or specification that verifying a program requires a "comparison." Indeed, Apple's proposed construction renders the claim language "at least" meaningless. Based on the foregoing, the Court will adopt Ancora's proposed construction of the disputed claim phrase:

Confirming whether a program is licensed using at least the verification structure.

F. Seventh Disputed Claim Term/ Phrase – All Asserted Claims

The parties dispute whether the claimed steps must be performed in a specific order to infringe on the '941 Patent. The parties' proposed constructions are shown below:

Apple's Proposed Construction	Ancora's Proposed Construction
The steps in each asserted claim must be performed in the order recited.	Unless the steps of a method actually recite an order, the claim is not limited to performance of the steps in the order recited.

"Unless the steps of a method actually recite an order, the steps are not ordinarily construed to require one. However, such a result can ensue when the method steps implicitly require that they be performed in the order written." *Altiris, Inc. v. Symantec Corp.*, 318 F.3d 1363, 1369-70 (Fed. Cir. 2003) (quoting *Interactive Gift Exp., Inc. v. Compuserve Inc.*, 256 F.3d 1323, 1342 (Fed. Cir. 2001)). "First, we look to the claim language to determine if, as a matter of logic or grammar, they must be performed in the order written. … If not, we next look to the rest of the specification to

United States District Court
Northern District of California

determine whether it 'directly or implicitly requires such a narrow construction.' If not, the sequence in which such steps are written is not a requirement." *Id.*

Apple argues that there is only one possible sequence to perform the steps in Claim 1, the order recited. "In this case, nothing in the claim or the specification directly or implicitly requires such a narrow construction." *Id.* As Ancora points out, Apple's proposed construction is contrary to the express teachings of the '941 Patent. In the Summary of the Invention, the '941 Patent teaches an embodiment of the invention in which the first step to be performed is setting up the verification structure. ('941 Patent 1:59-65.) In contrast to the express teachings of the '941 Patent, Apple argues that the first step is to select a program and the second step is setting up the verification structure. Nothing in the '941 Patent directly or implicitly requires that the steps be performed in the order recited.

Based on the foregoing analysis, the Court will not construe the claim to require the steps be performed in the order written.

IV. CONCLUSION

For the reasons set forth above, the Court provides the following construction of the disputed claim terms/phrases:

Disputed Claim Term/Phrase	**Construction**
volatile memory	Memory whose data is not maintained when the power is removed.
non-volatile memory	Memory whose data is maintained when the power is removed.
BIOS	An acronym for <u>B</u>asic <u>I</u>nput/<u>O</u>utput <u>S</u>ystem. It is the set of essential startup operations that run when a computer is turned on, which tests hardware, starts the operating system, and supports the transfer of data among hardware devices.
program	A set of instructions for software applications that can be executed by a computer.
license record	A record from a licensed program with information for verifying that licensed program.

verifying the program using at least the verification structure	Confirming whether a program is licensed using at least the verification structure.
All Asserted Claims	The steps of the Claim do not need to be performed in the order recited.

IT IS SO ORDERED.

Date: December 31, 2012

[illegible]

YVONNE GONZALEZ ROGERS
UNITED STATES DISTRICT COURT JUDGE

United States District Court
Northern District of California

US006411941B1

(12) **United States Patent**
Mullor et al.

(10) **Patent No.: US 6,411,941 B1**
(45) **Date of Patent: Jun. 25, 2002**

(54) METHOD OF RESTRICTING SOFTWARE OPERATION WITHIN A LICENSE LIMITATION

(75) Inventors: **Miki Mullor**; **Julian Valiko**, both of Ramat Hasharon (IL)

(73) Assignee: **Beeble, Inc.**, Newport Beach, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/164,777**

(22) Filed: **Oct. 1, 1998**

(30) **Foreign Application Priority Data**

May 21, 1998 (IL) 124571

(51) **Int. Cl.**^{7} **G06F 17/60**
(52) **U.S. Cl.** **705/59**; 705/50; 705/51; 705/53; 705/57
(58) **Field of Search** 705/51, 54, 56, 705/57, 58, 59, 1, 50, 52, 53; 713/187, 189, 200

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,866,769 A		9/1989	Karp	
4,903,296 A		2/1990	Chandra et al.	
4,924,378 A		5/1990	Hershey et al.	
5,386,369 A		1/1995	Christiano	
5,390,297 A		2/1995	Barber et al.	
5,479,639 A	*	12/1995	Ewertz et al.	395/430
5,490,216 A	*	2/1996	Richadson, III	380/4
5,671,412 A		9/1997	Christiano	
5,684,951 A	*	11/1997	Goodman et al.	395/188.01
5,754,763 A		5/1998	Bereiter	
5,758,068 A		5/1998	Brandt et al.	
5,758,069 A		5/1998	Olsen	
5,790,664 A		8/1998	Coley et al.	
5,826,011 A		10/1998	Chou et al.	
5,892,900 A	*	4/1999	Ginter et al.	395/186
5,905,860 A		5/1999	Olsen et al.	
6,000,030 A	*	12/1999	Steinberg et al.	713/200
6,006,190 A		12/1999	Baena-Arnaiz et al.	
6,021,438 A		2/2000	Duvvoori et al.	
6,023,763 A		2/2000	Grumpstrup et al.	
6,052,600 A	*	4/2000	Fette et al.	455/509
6,055,503 A		4/2000	Horstmann	
6,067,582 A	*	5/2000	Smith et al.	710/5
6,073,256 A		6/2000	Sesma	
6,078,909 A		6/2000	Knutson	
6,128,741 A		10/2000	Goetz et al.	
6,173,446 B1		1/2001	Khan et al.	
6,189,146 B1	*	2/2001	Misra et al.	717/11
6,192,475 B1		2/2001	Wallance	
6,198,875 B1	*	3/2001	Edenson et al.	386/94
6,226,747 B1		5/2001	Larsson et al.	
6,233,567 B1		5/2001	Cohen	
6,243,468 B1		6/2001	Pearce et al.	
6,272,636 B1		8/2001	Neville et al.	
6,298,138 B1		10/2001	Gotoh et al.	

FOREIGN PATENT DOCUMENTS

JP	408286906 A	*	11/1996	.... G06F/9/06

OTHER PUBLICATIONS

Dornbusch et al., Destop management software: no need to adjust your set., Infoworld, v17, n37, p60.*

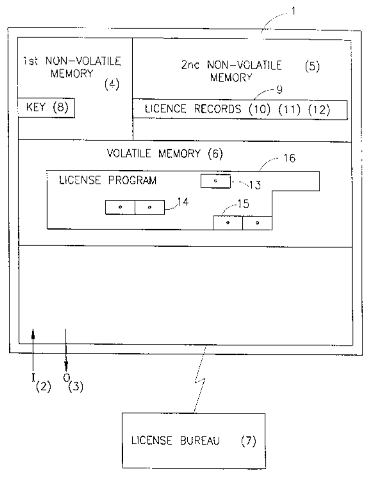
* cited by examiner

Primary Examiner—Hyung-Sub Sough
Assistant Examiner—Calvin L Hewitt
(74) *Attorney, Agent, or Firm*—Venable; Robert Kinberg; Jeffri A. Kaminski

(57) **ABSTRACT**

A method of restricting software operation within a license limitation that is applicable for a computer having a first non-volatile memory area, a second non-volatile memory area, and a volatile memory area. The method includes the steps of selecting a program residing in the volatile memory, setting up a verification structure in the non-volatile memories, verifying the program using the structure, and acting on the program according to the verification.

19 Claims, 2 Drawing Sheets


1
1st NON-VOLATILE MEMORY (4)
2nc NON-VOLATILE MEMORY (5)
9
KEY (8)
LICENCE RECORDS (10) (11) (12)
VOLATILE MEMORY (6)
16
LICENSE PROGRAM
13
14
15
I (2)
O (3)
LICENSE BUREAU (7)

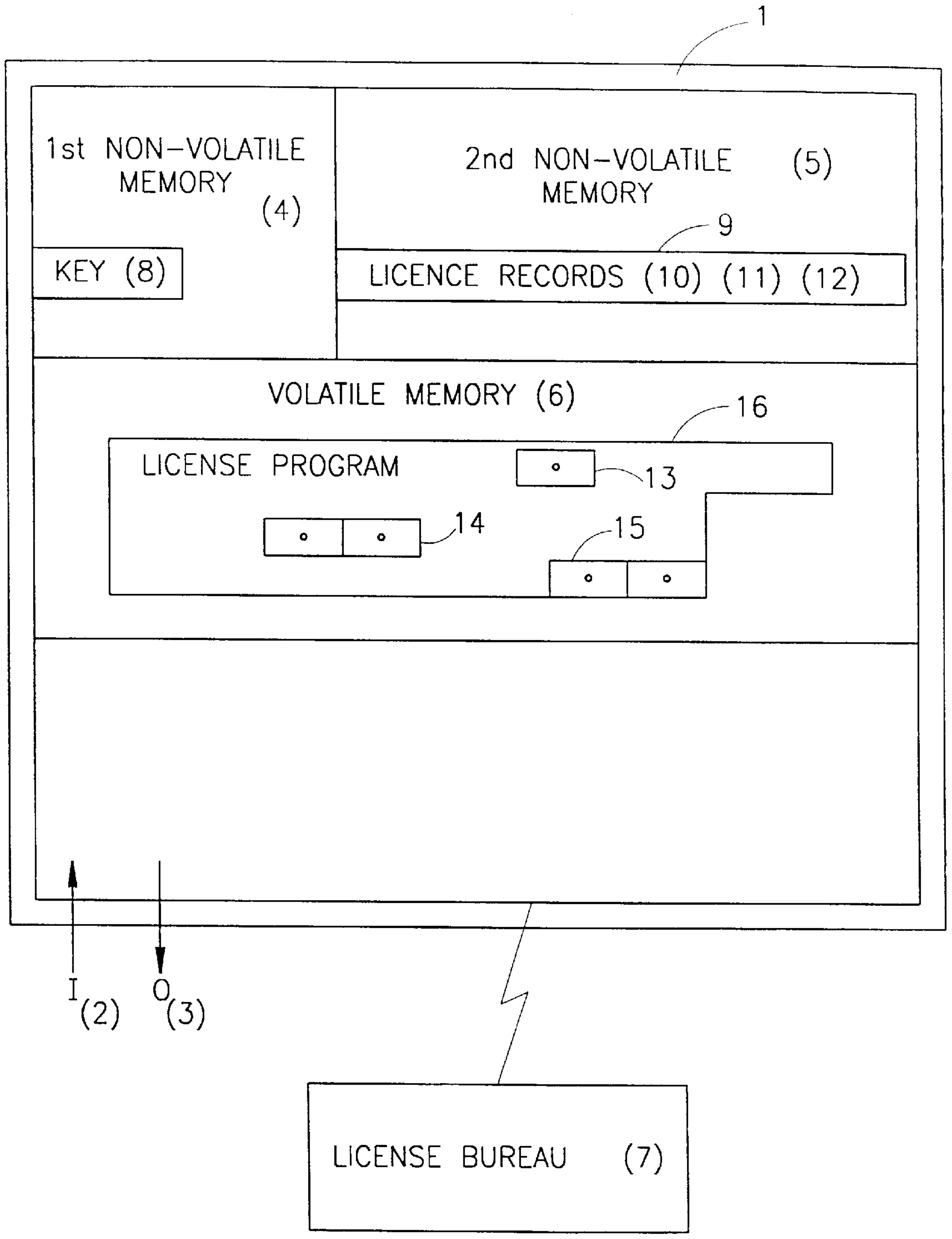

1
1st NON-VOLATILE MEMORY (4)
KEY (8)
2nd NON-VOLATILE MEMORY (5)
9
LICENCE RECORDS (10) (11) (12)
VOLATILE MEMORY (6)
16
LICENSE PROGRAM
13
14
15
I (2)
O (3)
LICENSE BUREAU (7)

FIG. 1

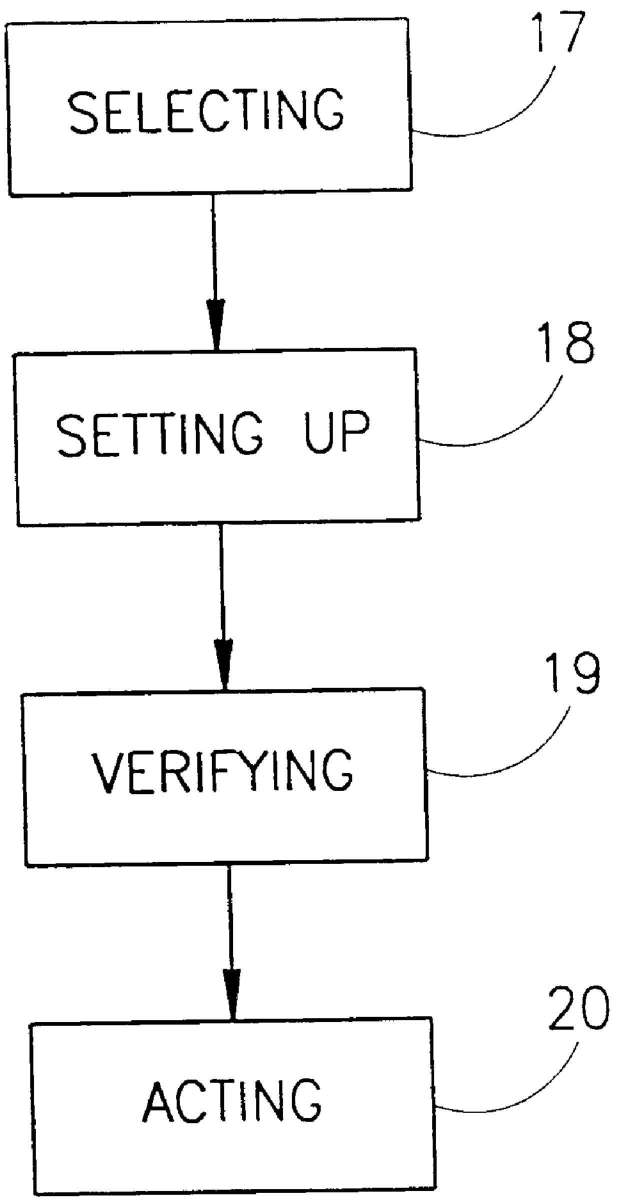


FIG.2

METHOD OF RESTRICTING SOFTWARE OPERATION WITHIN A LICENSE LIMITATION

FIELD OF THE INVENTION

This invention relates to a method and system of identifying and restricting an unauthorized software program's operation.

BACKGROUND OF THE INVENTION

Numerous methods have been devised for the identifying and restricting of an unauthorized software program's operation. These methods have been primarily motivated by the grand proliferation of illegally copied software, which is engulfing the marketplace. This illegal copying represents billions of dollars in lost profits to commercial software developers.

Software based products have been developed to validate authorized software usage by writing a license signature onto the computer's volatile memory (e.g. hard disk). These products may be appropriate for restricting honest software users, but they are very vulnerable to attack at the hands of skilled system's programmers (e.g. "hackers"). These license signatures are also subject to the physical instabilities of their volatile memory media.

Hardware based products have also been developed to validate authorized software usage by accessing a dongle that is coupled e.g. to the parallel port of the P.C. These units are expensive, inconvenient, and not particularly suitable for software that may be sold by downloading (e.g. over the internet).

There is accordingly a need in the art to provide for a system and method that substantially reduce or overcome the drawbacks of hitherto known solutions.

SUMMARY OF THE INVENTION

The present invention relates to a method of restricting software operation within a license limitation. This method strongly relies on the use of a key and of a record, which have been written into the non-volatile memory of a computer.

For a better understanding of the underlying concept of the invention, there follows a specific non-limiting example. Thus, consider a conventional computer having a conventional BIOS module in which a key was embedded at the ROM section thereof, during manufacture. The key constitutes, effectively, a unique identification code for the host computer. It is important to note that the key is stored in a non-volatile portion of the BIOS, i.e. it cannot be removed or modified.

Further, according to the invention, each application program that is to be licensed to run on the specified computer, is associated with a license record; that consists of author name, program name and number of licensed users (for network). The license record may be held in either encrypted or explicit form.

Now, there commences an initial license establishment procedure, where a verification structure is set in the BIOS so as to indicate that the specified program is licensed to run on the specified computer. This is implemented by encrypting the license record (or portion thereof) using said key (or portion thereof) exclusively or in conjunction with other identification information) as an encryption key. The resulting encrypted license record is stored in another (second) non-volatile section of the BIOS, e.g. E^2PROM (or the ROM). It should be noted that unlike the first non-volatile section, the data in the second non-volatile memory may optionally be erased or modified (using E^2PROM manipulation commands), so as to enable to add, modify or remove licenses. The actual format of the license may include a string of terms that correspond to a license registration entry (e.g. lookup table entry or entries) at a license registration bureau (which will be further described as part of the preferred embodiment of the present invention).

Having placed the encrypted license record in the second non-volatile memory (e.g. the E^2PROM), the process of verifying a license may be o commenced. Thus, when a program is loaded into the memory of the computer, a so called license verifier application, that is a priori running in the computer, accesses the program under question, retrieves therefrom the license record, encrypts the record utilizing the specified unique key (as retrieved from the ROM section of the BIOS) and compares the so encrypted record to the encrypted records that reside in the E^2PROM. In the case of match, the program is verified to run on the computer. If on the other hand the sought encrypted data record is not found in the E^2PROM database, this means that the program under question is not properly licensed and appropriate application define action is invoked (e.g. informing to the user on the unlicensed status, halting the operation of the program under question etc.)

Those versed in the art will readily appreciate that any attempt to run a program at an unlicensed site will be immediately detected. Consider, for example, that a given application, say Lotus 123, is verified to run on a given computer having a first identification code (**k1**) stored in the ROM portion of the BIOS thereof. This obviously requires that the license record (LR) of the application after having been encrypted using k1 giving rise to $(LR)_{k1}$ is stored in the E^2PROM of the first computer.

Suppose now that a hacker attempts to run the specified application in a second computer having a second identification code (k**2**) stored in the ROM portion of the BIOS thereof. All or a portion the database contents (including of course $(LR)_{k1}$) that reside in the E^2PROM portion in the first computer may be copied in a known per se means to the second computer. It is important to note that the hacker is unable to modify the key in the ROM of the second computer to K**1**, since, as recalled, the contents of the ROM is established during manufacture and is practically invariable.

Now, when the application under question is executed in the second computer, the license verifier retrieves said LR from the application and, as explained above, encrypts it using the key as retrieved from the ROM of the second computer, i.e k**2** giving rise to encrypted license record $(LR)_{k2}$. Obviously, the value $(LR)_{k2}$ does not reside in the E^2PROM database section of the second computer (since it was not legitimately licensed) and therefore the specified application is invalidated. It goes without saying that the data copied from the first (legitimate) computer is rendered useless, since comparing $(LR)_{k2}$ with the copied value $(LR)_{k1}$ results, of course, in mismatch.

The example above is given for clarity of explanation only and is by no means binding.

In its broadest aspect, the invention provides for a method of restricting software operation within a license limitation including; for a computer having a first non-volatile memory area, a second non-volatile memory area, and a volatile memory area; the steps of: selecting a program residing in the volatile memory, setting up a verification structure in the non-volatile memories, verifying the program using the structure, and acting on the program according to the verification.

An important advantage in utilizing non-volatile memory such as that residing in the BIOS is that the required level of system programming expertise that is necessary to intercept or modify commands, interacting with the BIOS, is substantially higher than those needed for tampering with data residing in volatile memory such as hard disk. Furthermore, there is a much higher cost to the programmer, if his tampering is unsuccessful, i.e. if data residing in the BIOS (which is necessary for the computer's operability) is inadvertently changed by the hacker. This is too high of a risk for the ordinary software hacker to pay. Note that various recognized means for hindering the professional-like hacker may also be utilized (e.g. anti-debuggers, etc.) in conjunction with the present invention.

In the context of the present invention, a "computer" relates to a digital data processor. These processors are found in personal computers, or on one or more processing cards in multi-processor machines. Today, a processor normally includes a first non-volatile memory, a second non-volatile memory, and data linkage access to a volatile memory. There are also processors having only one non-volatile memory or having more than two non-volatile memories; all of which should be considered logically as relating to having a first and a second non-volatile memory areas. There are also computational environments where the volatile memory is distributed into numerous physical components, using a bus, LAN, etc.; all of which should logically be considered as being a volatile memory area.

According to the preferred embodiment of the present invention, there is further provided a license authentication bureau which can participate in either or both of:

(i) establishing the license record in the second non-volatile memory; and

(ii) verifying if the key and license record in the non-volatile memory(s) is compatible with the license record information as extracted from the application under question.

The bureau is a telecommunications accessible processor where functions such as formatting, encrypting, and verifying may be performed. Performing these or other functions at the bureau helps to limit the understanding of potential software hackers; since they can not observe how these functions are constructed. Additional security may also be achieved by forcing users of the bureau to register, collecting costs for connection to the bureau, logging transactions at the bureau, etc.

According to one example of using the bureau, setting up a verification structure further includes the steps of: establishing, between the computer and the bureau, a two-way data-communications linkage; transferring, from the computer to the bureau, a request-for-license including an identification of the .computer and the license-record's contents from the selected program; forming an encrypted license-record at the bureau by encrypting parts of the request-for-license using part of the identification as the encryption key; and transferring, from the bureau to the computer, the encrypted license-record.

According to another example of using the bureau, verifying the program further includes the steps of: establishing, between the computer and the bureau, a two-way data-communications linkage; transferring, from the computer to the bureau, a request-for-license-verification including an identification of the computer, the encrypted license-record for the selected program from the second non-volatile memory, and the licensed-software-program's license-record contents; enabling the comparing at the bureau; and transferring, from the bureau to the computer, the result of the comparing.

The actual key that serves for identifying the computer may be composed of the pseudo-unique key exclusively, or, if desired, in combination with information, e.g. information related to the registration of the user such as e.g. place, telephone number, user name, license number, etc. In the context of the present invention, a "pseudo-unique" key may relate to a bit string which uniquely identifies each first non-volatile memory. Alternately the "pseudo-unique" key may relate to a random bit string (or to an assigned bit string) of sufficient length such that: there is an acceptably low probability of a successful unauthorized transfer of licensed software between two computers, where the first volatile memories of these two computers have the same key.

It should be noted that the license bureau might maintain a registry of keys and of licensed programs that have been registered at the bureau in association with these keys. This registry may be used to help facilitate the formalization of procedures for the transfer of ownership of licensed software from use on one computer to use on another computer.

Constructing the key in the manner specified may hinder the hacker in cracking the proposed encryption scheme of the invention, in particular when the establishment of the license record or the verification thereof is performed in the bureau. Those versed in the art will readily appreciate that the invention is by no means bound by the data, the algorithms, or the manner of operation of the bureau. It should be noted that the tasks of establishing and/or verifying a license record may be shared between the bureau and the computer, done exclusively at the computer, or done exclusively at the bureau. The pseudo-unique key length needs to be long enough to hinder encryption attack schemes. The establishing of the key may be done at any time from the non-volatile memory's manufacture until an attempted use of an established license-record in the non-volatile memory. The key is used for encryption or decryption operations associated with license-records. In principle, the manufacturer of the licensed-software-program may specify the license-record format and therefore different formats may, if desired, be used for respective applications.

According to the preferred embodiment of the present invention, the pseudo-unique key is a unique-identification bit string that is written onto the first non-volatile memory by the manufacturer of the is memory media.

According to one, non-limiting, preferred embodiment of the present invention, the first non-volatile memory area is a ROM section of a BIOS; the second non-volatile memory area is a E^2PROM section of a BIOS; and the volatile memory is a RAM e.g. hard disk and/or internal memory of the computer.

The present invention also relates to a non-volatile memory media used as a BIOS of a computer, for restricting software operation within a license limitation, wherein a pseudo-unique key is established.

According to the preferred embodiment of the non-volatile memory media of the present invention, the pseudo-unique key is established in a ROM section of the BIOS.

BRIEF DESCRIPTION OF THE DRAWINGS

In order to understand the invention and to see how it may be carried out in practice, a preferred embodiment will now be described, by way of non-limiting example only, with reference to the accompanying drawings, in which:

FIG. **1** is a schematic diagram of a computer and a license bureau; and

FIG. **2** is a generalized flow chart of the sequence of operations performed according to one embodiment of the invention.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

A schematic diagram of a computer and a license bureau is shown in FIG. **1**. Thus, a computer processor **(1)** is associated with input operations **(2)** and with output operations **(3)**. This computer (processor) internally contains a first non-volatile memory area **(4)** (e.g. the ROM section of the BIOS), a second non-volatile memory area **(5)** (e.g. the E^2PROM section of the BIOS), and a volatile memory area **(6)** (e.g. the internal RAM memory of the computer).

The computer processor is in temporary telecommunications linkage with a license bureau **(7)**.

The first non-volatile memory includes a pseudo-random identification key **(8)**, which exclusively or in combination with other information (e.g. user name), is sufficient to uniquely differentiate this first non-volatile memory from all other first non-volatile memories. As specified before, said key constitutes unique identification of the computer.

The second non-volatile memory includes a license-record-area **(9)** e.g. which contains at least one encrypted license-record (e.g. three records **10–12**). The volatile memory accommodates a license program **(16)** having license record fields **(13–15)** appended thereto. By way of example said fields stand for Application names (e.g. Lotus 123), Vendor name (Lotus inc.), and number of licensed copies (1 for stand alone usage, >1 for number of licensed users for a network application).

Those versed in the art will readily appreciate that the license record is not necessarily bound to continuous fields. In fact, the various license content components of the data record may be embedded in various locations in the application. Any component may, if desired, be encrypted.

Each one of the encrypted license records **(10–12)** is obtained by encrypting the corresponding license record as extracted from program **16**, utilizing for encryption the identification key **(8)**.

In a typical, yet not exclusive, sequence of operation, a transaction/request is sent, by the computer to the bureau. This transaction includes the key **(8)**, the encrypted license-records **(10–12)**, contents from the license program used in forming a license record (e.g. fields **13–15**), and other items of information as desired.

The bureau forms the proposed license-record from the contents, encrypts (utilizing predetermined encryption algorithm) the so formed license-record using the key **(8)**, and compares the so formed encrypted license-record with the license-record **(10–12)**. The bureau generates an overlay according to the result of the comparison indicating successful comparison, non-critical failure comparison and the critical failure comparison.

The bureau returns the overlay which will direct the computer in subsequent operation. Thus, a success overlay will allow the license program to operate. A non-critical failure overlay will ask for additional user interactions. A critical failure overlay will cause permanent disruption to the computer's BIOS operations. Thus, software operation of the program is methodologically according to a license limitation restriction.

Those versed in the art will readily appreciate that the implementation as described with reference to FIG. **1** is by

no means binding. Thus, by way of non-limiting example, the bureau, instead of being external entity may form part of the computer.

Attention is now directed to FIG. **2**, showing a generalized flow chart of the sequence of operations performed according to one embodiment of the invention.

Thus, selecting **(17)** a program includes the step of: establishing a licensed-software-program in the volatile memory of the computer wherein the licensed-software-program includes contents used to form a license-record. These contents, be they centralize or decentralized, may include terms, identifications, specifications, or limitations related to the manufacturer of a software product, the distributor of a software product, the purchaser of a software product, a licensor, a licensee, items of computer hardware or components thereof, or to other terms and conditions related to the aforesaid.

Setting up **(18)** the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in the first non-volatile memory area; and establishing at least one license-record location in the first or the second nonvolatile memory area.

Establishing a license-record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and establishing the encrypted license-record in one of the at least one established license-record locations (e.g. **10–12** in FIG. **1**).

Verifying **(19)** the program includes the steps of: encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the first or the second non-volatile memory area, using the key; and comparing the encrypted licensed-software-program's license-record contents with the encrypted license-record in the first or the second non-volatile memory area, or comparing the licensed-software-program's license-record contents with the decrypted license-record in the first or the second non-volatile memory area.

Acting **(20)** on the program includes the step of: restricting the program's operation with predetermined limitations if the comparing yields non-unity or insufficiency. In this context "non-unity" relates to being unequal with respect to a specific equation (e.g. A=B+1); and "insufficiency" relates to being outside of a relational bound (e.g. A>B+1). "Restricting the program's operation with predetermined limitations" may include actions such as erasing the software in volatile memory, warning the license applicant/user, placing a fine on the applicant/user through the billing service charges collected at the license bureau (if applicable), or scrambling sections of the BIOS of the computer (or of functions interacting therewith).

The present invention has been described with a certain degree of particularity but it should be understood that various modifications and alterations may be made without departing from the scope or spirit of the invention as defined by the following claims.

What is claimed is:

1. A method of restricting software operation within a license for use with a computer including an erasable, non-volatile memory area of a BIOS of the computer, and a volatile memory area; the method comprising the steps of:

selecting a program residing in the volatile memory,

using an agent to set up a verification structure in the erasable, non-volatile memory of the BIOS, the verification structure accommodating data that includes at least one license record, verifying the program using at least the verification structure from the erasable non-volatile memory of the BIOS, and

acting on the program according to the verification.

2. A method according to claim **1**, further comprising the steps of:

establishing a license authentication bureau.

3. A method according to claim **2**, wherein setting up a verification structure further comprising the steps of: establishing, between the computer and the bureau, a two-way data-communications linkage; transferring, from the computer to the bureau, a request-for-license including an identification of the computer and the license-record's contents from the selected program; forming an encrypted license-record at the bureau by encrypting parts of the request-for-license using part of the identification as an encryption key; transferring, from the bureau to the computer, the encrypted license-record; and storing the encrypted license record in the erasable non-volatile memory area of the BIOS.

4. A method according to claim **2**, wherein verifying the program further comprises the steps of: establishing, between the computer and the bureau, a two-way data-communications linkage; transferring, from the computer to the bureau, a request-for-license verification including an identification of the computer, an encrypted license-record for the selected program from the erasable, non-volatile memory area of the BIOS, and the program's license-record; enabling the comparing at the bureau; and transferring, from the bureau to the computer, the result of the comparing.

5. A method according to claim **3** wherein the identification of the computer includes the unique key.

6. A method according to claim **1** wherein selecting a program includes the steps of: establishing a licensed-software-program in the volatile memory of the computer wherein said licensed-software-program includes contents used to form the license-record.

7. A method according to claim **6** wherein using an agent to set up the verification structure includes the steps of: establishing or certifying the existence of a pseudo-unique key in a first non-volatile memory area of the computer; and establishing at least one license-record location in the first nonvolatile memory area or in the erasable, non-volatile memory area of the BIOS.

8. A method according to claim **6** wherein establishing a license-record includes the steps of: forming a license-record by encrypting of the contents used to form a license-record with other predetermined data contents, using the key; and establishing the encrypted license-record in one of the at least one established license-record locations.

9. A method according to claim **7** wherein verifying the program includes the steps of: encrypting the licensed-software-program's license-record contents from the volatile memory area or decrypting the license-record in the erasable, non-volatile memory area of the BIOS, using the pseudo-unique key; and comparing the encrypted licenses-software-program's license-record contents with the encrypted license-record in the erasable, non-volatile memory area of the BIOS, or comparing the license-software-program's license-record contents with the decrypted license-record in erasable non-volatile memory area of the BIOS.

10. A method according to claim **9** wherein acting on the program includes the step: restricting the program's operation with predetermined limitations if the comparing yields non-unity or insufficiency.

11. A method according to claim **1** wherein the volatile memory is a RAM.

12. The method of claim **1**, wherein a pseudo-unique key is stored in the non-volatile memory of the BIOS.

13. The method of claim **1**, wherein a unique key is stored in a first non-volatile memory area of the computer.

14. The method according claim **13**, wherein the step of using the agent to set up the verification record, including the license record, includes encrypting a license record data in the program using at least the unique key.

15. The method according to claim **14**, wherein the verification comprises:

extracting the license record from the software program;

encrypting the license record using the unique key stored in the first non-volatile memory area of the computer to form second encrypted license information; and

comparing the encrypted license information stored in the erasable, non-volatile memory area of the BIOS of the computer with the second encrypted license information.

16. The method according to claim **13**, wherein the step of verifying the program includes a decrypting the license record data accommodated in the erasable second non-volatile memory area of the BIOS using at least the unique key.

17. The method according to claim **13**, wherein the step of verifying the program includes encrypting the license record that is accommodated in the program using at least the unique key.

18. A method for accessing an application software program using a pseudo-unique key stored in a first non-erasable non-volatile memory area of a computer, the first non-volatile memory area being unable to be programmatically changed, the method, comprising:

loading the application software program residing in a non-volatile memory area of the computer;

using an agent to perform the following steps:

extracting license information from software program;

encrypting license information using the pseudo-unique key stored in the first non-volatile memory area;

storing the encrypting license information in a second erasable, writable, non-volatile memory area of the BIOS of the computer;

subsequently verifying the application software program based on the encrypted license information stored in the second erasable, writable, non-volatile memory area of the BIOS; and

acting on the application software program based on the verification.

19. The method of claim **18**, wherein the verification comprises:

extracting the license information from the software program;

encrypting the license information using the pseudo-unique key stored in the first non-volatile memory area of the computer to form second encrypted license information; and

comparing the encrypted license information stored in the second erasable, writable, non-volatile memory area of the BIOS of the computer with the second encrypted license information.

* * * * *

CERTIFICATE OF ELECTRONIC SERVICE

I hereby certify that on July 2, 2013, I electronically filed the foregoing **APPEAL BRIEF OF PLAINTIFF-APPELLANT, ANCORA TECHNOLOGIES, INC.** with the Clerk for the Federal Circuit Court of Appeals using the ECF System which will send notification to the participants of the ECF System as listed on the Court's Notice of Docket Activity.

BROOKS KUSHMAN P.C.

/s/ John S. LeRoy
Mark A. Cantor
John S. LeRoy
Marc Lorelli
John P. Rondini
BROOKS KUSHMAN P.C.
1000 Town Center
Twenty-Second Floor
Southfield, Michigan 48075-1238
(248) 358-4400

Attorneys for Plaintiff-Appellant

CERTIFICATE OF COMPLIANCE PURSUANT TO FED. R. APP. P. 32(a)(7)(B)

This brief complies with the type-volume limitation of Federal Rule of Appellate Procedure 32(a)(7)(B). The brief contains 3,678 words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(a)(7)(B)(iii).

This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type style requirements of Federal Rule of Appellate Procedure 32(a)(6). The brief has been prepared in a proportionally spaced typeface using Microsoft® Word 2010 and Times New Roman typeface, 14-point.

Respectfully submitted,

BROOKS KUSHMAN P.C.

/s/ John S. LeRoy

Mark A. Cantor
John S. LeRoy
Marc Lorelli
John P. Rondini
BROOKS KUSHMAN P.C.
1000 Town Center
Twenty-Second Floor
Southfield, Michigan 48075-1238
(248) 358-4400

Attorneys for Plaintiff-Appellant

Date: July 2, 2013